Frontend exam topics

2022-04-26 version by JV

Frontend exam, ideas on what to study

==========

1. **React**: a JavaScript SPA UI building library

- renders the output page (HTML+CSS+JS) DOM based on
  - your page template code (public>index.html+ React components inside other React components) AND
  - the data provided by AJAX (with or without Redux)

2. **JSX**: Basics *https://www.youtube.com/watch?v=FO4couzuJIk&t=67s (from 1:07 to **14:27**)*

- XML-like language mixing JS and React component markup, not XML as it's not following XML rules, nor HTML structure rules either.

- One syntax example: What is the following?

var a =123;

<Xyz abc={{a}} /> // first go to JS mode with { }, then create an object {a}, (thus same as ={{a:a}})

*- React components with **C**apitalFirstLetter, HTML elements/attributes with small letter*

*- **className(s)** attribute or so (in React, react-rendered) vs. **class** attribute (HTML, already ready html. Thus, being able to mix JSX and ready html, only if unavoidable)*

- when returning JSX, wrap it inside ( ) ((((Or make sure to start JSX from same line as the return keyword)))

3. **React Hooks**: Basics

a. *React hooks basics (recap also the ES object destructor assignment if needed)*

https://youtu.be/mxK8b99iJTg?t=40 // Just forget the pre-release react installation, no need for "npm i ..."

- from 00:40 to 19:50, 19 mins // Simple output and input UI with react hooks. (No AJAX/persistence here)

((https://reactjs.org/docs/hooks-intro.html))

https://youtu.be/dpw9EHDh2bM?t=1061

- from 17:45 until 54:13 = <36 mins NOTICE, NOT THE LATTER PART OF THE SHOW!

- Here anything with class/constructor you can just follow, and only fully understand the hooks version of the same.

- https://www.youtube.com/watch?v=TNhaISOUy6Q

*-from 0:00 **until 6:58**, our 3 Hooks (useState, useEffect, useContext) shown in just 7 mins, but going fast. Pause and read code!*

https://reactjs.org/docs/hooks-overview.html

https://reactjs.org/docs/hooks-state.html

b. *Effect Hooks*

https://reactjs.org/docs/hooks-effect.html

- useEffect inner function which HAS replaced the React component life-cycle event handlers (Where you can 'attach' your event-handler functions). inner function, function defined inside the component function. So, the visibility of the function is inside the outer function only.

c. *General rules of Hooks*

https://reactjs.org/docs/hooks-rules.html

d. *Custom Hooks (a bit Advanced topic)*

https://reactjs.org/docs/hooks-custom.html

- custom hooks, like the ones in the second video:

useWindowWidth

useDocumentTitle

e. *Context Hooks (Advanced topic, useful still in our case)*

https://reactjs.org/docs/hooks-reference.html#usecontext
https://www.youtube.com/watch?v=lhMKvyLRWo0

4. create-react-app

The tool & dev project environment it creates: Basic understanding.

- What kind of project create-react-app creates? (npm start => dev time environment with Node server and e.g. React Dev tools)

- What's the relationship with the /public/index.html and the React app? How the React app starts and builds up the page?

- How is it related to the build version = when published and put to 'production environment'?

(npm build => /build folder with only few mashed up .html and .js (and needed .css plus other static files), no Node anymore, no ES5,6,7,8, just browser runnable JS, served to the client's web browser by even a static web server, from ports 80/443 or so)

5. SPA = Single-Page Application

Only one web page downloaded from the Web Server, but then with JS & AJAX that single page's DOM updated constantly. Showing/hiding certain Views so that it looks like we would have several Pages

- SPA in general. Single page which is changed based on user actions, by JS code, AJAX requests/responses and react-router routing "going to a new View" with possible routing parameters.

- React components can be Views = SPA pages we can get routed to

- Some others are re-usable children of the Views

- Some are container components who have/fetch the data

- Some others are presentational components who get the data from parent, and who only show what they get (plus possibly provide links/buttons related to _that_ item that it's showing)

6. React routing

SPA front-end routing between the Views (~like “going” to a different page)

- though really just showing and hiding React Views
- we can also send parameter data while going to another View, e.g. id:s

- How is the react-router routing working in general. Routing parameters?

https://youtu.be/Law7wfdg_ls?t=73

- from 1:13 (link above already at that time) until 16:46 at least.

- after 16:46 starts "customer routing" i.e. routing with parameters. Maybe that too, if have time. 33 mins in total.

7. Theming

- How is the theme ((((and redux store. NOT THIS TIME)))) shared with / provided to all React components? (Answer: Injected to the root React element, and theme-abled child components used.

8. AJAX

- Something about AJAX? (Do we get the result out as JSON text, or already auto-parsed to be a JS object)

9. LocalStorage

Browsers are able to save text files to the computer’s disk and open them with a key/name e.g. days later. If objects are stringified=(serialized as JSON text) we can even persist (data) objects.

10. Full-stack open 202X, the reading list

- has now the green parts that are interesting from Frontend learning point of view!

https://github.com/valju/docs_backend_design/blob/master/FSO/FSOReadingList.pdf (in Backend docs repo) Note: some things from part 7 might be added for the Spring 2022 exam and onwards!

11. For thoughts: Possible frontend project design & creation steps

https://github.com/valju/docs_frontend_design/blob/master/FrontendRelatedSteps_SimilarToOurCases.pdf

~~12. For thoughts: Used tech in an example Frontend project~~

~~*https://github.com/valju/docs_frontend_design/blob/master/Frontend_UsedTech_2021.pdf*~~

~~13. Extra: An 'example' Frontend project (forget Redux though)~~

~~*https://github.com/valju/idea-case-frontend*~~

NOT TO EXAM THIS TIME FROM THIS ON

==========

NOT TO EXAM THIS TIME FROM THIS ON

==========

Redux

Front-end side (=in browser memory) State management.

- part of the data model temporarily kept in browser memory (in Redux store/state) in the frontend
- frequency of updates from backend depends on the nature of the data. List of selectable colors for some product maybe not before next login, messages maybe polled and fetched by frontend (or pushed by backend) a lot more often.
- many components can share the same data model
- **Redux used to be the way/thing to do at least 2-3 years ago. Not sure what is situation now in 5/2022?**.

React-Redux

React components will be data- and event-bound to Redux store.

- actions dispatched to Redux code.
- redux store state bound to the React state of the components

React-Material-UI

Material-UI styled React components that share common theme and styles

- Those React components need to also be 1) Redux-connected and 2) React-routed!